Communication Optimizations for a Wireless Distributed Prognostic Framework

Sankalita Saha
RIACS/NASA Ames Research Center
Moffett Field, CA 94035
ssaha@riacs.edu

Bhaskar Saha
MCT/NASA Ames Research Center
Moffett Field, CA 94035
bhaskar.saha@nasa.gov

Kai Goebel
RIACS/NASA Ames Research Center
Moffett Field, CA 94035
kai.goebel@nasa.gov

Abstract—Distributed architecture for prognostics is an essential step in prognostic research in order to enable feasible real-time system health management. Communication overhead is an important design problem for such systems. In this paper we focus on communication issues faced in the distributed implementation of an important class of algorithms for prognostics – particle filters. In spite of being computation and memory intensive, particle filters lend well to distributed implementation except for one significant step – resampling. We propose new resampling scheme called *parameterized resampling* that attempts to reduce communication between collaborating nodes in a distributed wireless sensor network. Analysis and comparison with relevant resampling schemes is also presented. A battery health management system is used as a target application. [1] [2]

TABLE OF CONTENTS

1. INTRODUCTION

As health management issues become an increasingly dominant concern in the aerospace domain, it becomes imperative to explore efficient architectures for system implementation. An important constituent of a health management system is prognostics or determining remaining useful life. Such systems consist of multiple sensors that monitor various subsystems; the data collected from these sensors are processed by suitable algorithms to determine the health of the system. Thus, they are complex and deploy sophisticated algorithms as well as sensor instrumentation.

Most of the system development assumes a centralized health management architecture, i.e., a central computing machine collects all the sensor data, processes them, and then runs various diagnostic and prognostic algorithms. Such a system architecture has to deal with several challenges: (a) large amounts of sensor data being collected for more refined analysis (e.g., high frequency vibration data for structures health management or data with high sampling rate for avionics health management); (b) increasingly more complex algorithms – intensive in terms of memory as well as computation speed – are being deployed to process this data, exceeding the capabilities of single-processor systems; and (c) vulnerability to complete loss of functionality in case of a crash of the central processor/monitor. In the last case, considerable amount of time and effort is required to recover and restore back the health management system and in many instances such a recovery may not be possible at all. A centralized architecture is not sufficient for the increasingly multi-operational and complex systems of today.

Distributed health management is the next step in the evolution of prognostic methodologies. A distributed architecture comprises multiple smart sensor devices that monitor different parts of a system and collaborate when computationally intensive prognostic algorithms or large amounts of data are involved that cannot be handled efficiently by a single processor/node. Recent advances in smart sensor technology combining the power of embedded computing devices with sensors and wireless transmission technology make the practical implementation of such systems feasible.

One of the most important design considerations in such systems is the communication overhead. Inefficient communication architecture can reduce computational performance gains obtained from task distribution. Additionally, design issues faced in a wired distributed system are significantly different from that faced in wireless systems. Thus, prognostics algorithms need to be specially designed for such systems to increase their communication efficiency.

Particle filters provide an important class of algorithms employed in system prognostics. They are computation and memory intensive algorithms but lend very well to distributed implementations except for one significant step – resampling. Efforts for efficient distributed resampling schemes have been made. However, the suitability of such schemes is heavily dependent on the application as well as target implementation platform. In this paper, we explore different resampling schemes for a particle filter based battery health management system with special focus on

[1] U.S. Government work not protected by U.S. copyright.
[2] IEEEAC paper #1332, Version 3, Updated November 2, 2008

reduction of communication between collaborating nodes in a wireless sensor network. In addition, we propose a new resampling scheme for particle filter systems targeted towards reducing the communication overhead. The specific target platform in our case is a network of Sun Microsystems SPOT (Small Programmable Object Technology) devices. However, the techniques developed in this paper are generic enough for use in other particle filter based systems.

2. BACKGROUND

The field of prognostics is still maturing and hence significant work in distributed prognostics does not exist. A few efforts have been made recently. The authors briefly outline a distributed prognostics system architecture in [1] where tasks are distributed at the prognostics algorithm level, i.e., identifying the different system modules and where they fit into a given system using prognostics. In [2] a distributed network of smart sensor elements integrated using a knowledge-driven environment to perform self-diagnosis of health and participate in a hierarchy of health determination at sensor, process, and system levels. This network will be used as an element of the prototype intelligent rocket test facility being implemented at NASA Stennis Space Center. In [3] a hardware multi-cellular sensing and communication network (a smart "skin") is presented and discussed for health management of space vehicles. The main aim of such a smart "skin" aim is to detect and react to impacts caused by projectiles that, for a vehicle in space, might be micro-meteoroids or space debris.

Some of the techniques used in prognostics – such as particle filters – have been investigated in the context of distributed implementations. For example, in [4] the authors present three different distributed methods for implementing particle filter system. However, the algorithms presented do not distribute the algorithm fully. In [5] the authors present a parallel particle filter implementation on a shared-memory multiprocessor cluster. Sensor networks have gained popularity of late and often employ particle filters for tracking objects. Distributed particle filters for such applications have also been explored ([6], [7], [8], [9], [10]). However, none of the work above addresses the problem of communication overhead involved in distributed particle filters and improving the communication efficiency.

Communication issues are widely recognized and analyzed in the context of generic distributed networks. It is most often the highest contributor to resource management costs, typically higher by orders of magnitude as compared to other factors. Various approaches have been explored to mitigate the effects of communication issues. For example, an approximate dynamic programming approach that integrates the value of information and the cost of transmitting data over a rolling time horizon is presented in [11] in the context of object tracking with a distributed sensor network. However, the above technique is specific to a given application domain and may not be easily extended to other domains. Network topology can play an important role in improving communication overheads, and in [12] the authors present few recent developments in networking techniques for multiple sensor systems. In [13], the problem of minimizing communication in general distributed systems is considered in a discrete-event formalism where the system is modeled as a finite-state automaton. This work provides an interesting approach; however it focuses more on an analysis framework.

As may be observed from the above discussed work, there is a distinct lack of work on design of distributed algorithms in the context of prognostics that focus on reducing the communication overhead. The work presented in this paper attempts to address this problem with focus on a special class of algorithms – particle filters. However, the solutions presented are generic enough for extension to other application domains that are particle-filter based.

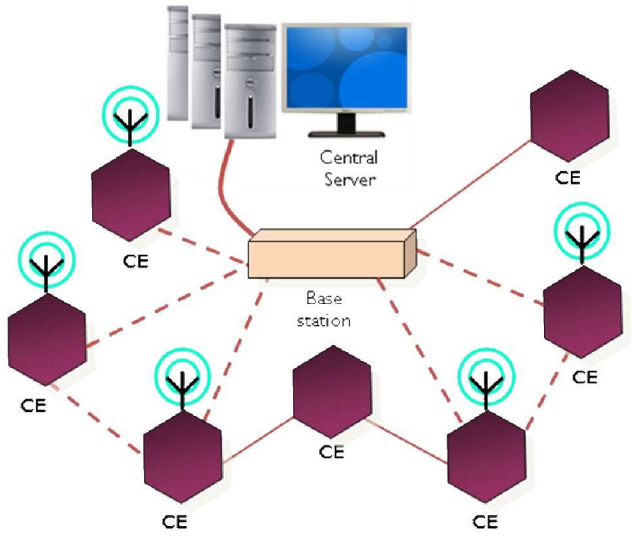


Figure 1 – Overview of distributed prognostics system architecture. Note that all the CEs may not have wireless connectivity.

3. DISTRIBUTED PROGNOSTICS

Overview

In a distributed prognostics system multiple smart sensor devices are employed that monitor various subsystems or modules. The devices perform diagnostics operations and trigger the prognostics mode based on user defined thresholds and rules. An example of such a distributed prognostics system is shown in Figure 1. As shown in this figure, the system architecture in general comprises multiple computing elements (*CE*s) each of which consists of a sensor or a set of sensors and a communication device i.e., a wireless transreceiver or wired communication capabilities besides an embedded processing element. Though, in this paper we mainly focus on a wireless implementation

platform, in a real-life scenario the connection between the sensor devices may be wired or wireless. A wireless connection enables more flexibility in the system design with regards to placement of sensors. However, this flexibility comes at the cost of significantly higher communication overhead that involves concerns such as synchronization and packet losses besides performance issues.

Under most operating conditions the CEs would use both the sensor and embedded processing capabilities. However, in many cases their sensor capabilities may not be utilized, i.e., they could act as monitors for the rest of the system – schedule tasks, detect failures and initiate recovery, provide access to resources such as an external database etc. – or act as "helpers" to offload the computation requirements from other CEs in order to maintain real-time constraints of the application.

There are two operating modes for a *CE*: *diagnostics* and *prognostics* [14]. A *CE* runs in the default mode of diagnostics until a flag is raised by some *CE*. Depending on the current state (i.e., availability of resources) it then switches to prognostics mode. Thus, in the prognostics mode it is not necessary that all the *CE*s are utilized; some of them may be busy monitoring critical components or may not have enough computing power to simultaneously execute default operations. Note that the diagnostics operations are not halted in the prognostics mode. To ensure that a *CE* can support such multi-tasking efficiently the prognostics algorithms need to be distributed efficiently.

In Figure 1, the *basestation* is not statically determined. Initially, a default *basestation* is allocated whose main job is to monitor the *CE*s and coordinate information exchange. This information exchange not only involves the *CE*s themselves but also includes entities such as the user, a database server (for accessing history knowledge, or store collected sensor data for later analysis) and other clusters of *CE*s in a hierarchical system. When the prognostics mode is triggered, either the *base station* or the *CE* that triggered the mode makes an estimate of available computing resources. The new *basestation* is chosen which then partitions, schedules and delegates tasks accordingly.

Implementation Platform

The implementation platform consists of a network of smart sensor devices from Sun Microsystems called SPOT (Small Programmable Object Technology) devices. The SPOT device is a small, wireless, battery powered experimental platform which is built by stacking a Sun SPOT processor board with a sensor board and battery. The sensor board includes a range of built-in sensors as well as the ability to easily interface to external devices. In terms of processing power, each Sun SPOT has a 180MHz 32-bit ARM920T core processor with 512K RAM and 4M Flash. As shown in [14] this provides sufficient multi-tasking capabilities for the systems under consideration in this work.

The SPOT devices communicate using radio channels. The processor board has a 2.4GHz radio with an integrated antenna on the board. The radio is IEEE 802.15.4 compliant. The communication capability of this system was severely overloaded for our distributed prognostics implementation [14] due to limitations imposed by the restrictions on the message length by the communication channel. In the context of a particle-filter based prognostics system, this posed a significant design challenge, as particle filters are data intensive algorithms. Particle filters main system information in terms of states and each state is represented using large number of as samples (further details are provided in section 4). Due to the above-mentioned communication limitations, the full state information for a single particle filter iteration could not be packed into one message. Thus, the message had to be broken into multiple parts and sent iteratively as separate messages. This increased the amount of time spent in communication considerably, as the time to set up and send a message as well as receiving the message is significantly high. Furthermore, communication was acknowledgement-based in order to handle lost or corrupt messages, thereby further adding to the communication overhead.

It may be emphasized that such problems with high volume of communication are commonly encountered in many distributed and wireless systems. Therefore, there is a strong need to design algorithms which reduce both the number of communication messages as well as the message lengths. Battery power consumption gets significantly affected by wireless communication usage as well. The maximum capacity of the built-in battery (3.7V rechargeable, Lithium-ion battery) for the SPOT devices is 720 mAhr. Since battery power management is a key issue in wireless systems, this provides further motivation for improving communication efficiency.

4. DISTRIBUTED PARTICLE FILTERS

Particle filters (PFs) provide a powerful technique for prognostics. They are based on Bayesian learning networks and essentially implement a recursive Bayesian filter using Monte Carlo (MC) simulations; hence they are also known as sequential MC methods. In the prognostics domain, PFs are mainly used to track progression of system state in order to make estimations of remaining useful life (RUL).

Bayesian techniques provide a general rigorous framework for such dynamic state estimation problems where the core idea is to construct a probability density function (pdf) of the state based on all available information. For the PF approach ([15], [16]) this is done by approximating the pdf with a set of particles (points) representing sampled values from the unknown state space, and a set of associated weights denoting discrete probability masses. The particles are generated and recursively updated from a nonlinear process model that describes the evolution in time of the system under analysis, a measurement model, a set of

available measurements and an a priori estimate of the state pdf.

Particle filter methods assume that the state equations can be modeled as a first order Markov process with the outputs being conditionally independent. This can be represented as follows:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \omega_k$$
$$\mathbf{y}_k = h(\mathbf{x}_k) + v_k \quad (1)$$

where, k is the time index, $\mathbf{x}$ denotes the state, $\mathbf{y}$ is the output or measurements, and both ω and v are samples from noise distributions. The algorithm is initiated by a best guess estimate of the state space represented as a set of N weighted particles $\{(w_k^{(i)}, \mathbf{x}_k^{(i)}): i=1,\ldots,N\}$. The importance weights $w_k^{(i)}$ are approximations to the relative posterior probabilities of the particles.

In terms of computation, a particle filter based system consists of the following three computational steps:

1. Sampling: Generation of samples (particles) of the unknown state based on the given sampling function to provide an estimate of the current state of the system and also propagate the particles from the previous time step to the current time using Eq. (1).
2. Weight Calculation: Update importance weights for each particle based on external observations:

$$w_k^{(i)} = w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{x}_{k-1})}{\pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})}. \quad (2)$$

3. Resampling: Redrawing particles from the same probability density based on some function (resampling function) of the particle weights such that the weights of the new particles are approximately equal.

Though particle filters are expensive with respect to computation as well as memory requirements, they exhibit considerable amount of data parallelism to enable distributed processing. All the steps enlisted above except resampling can be completely distributed over independently executing *CE*s. Since there are no data dependencies during sampling and weight updates, these segments of particle filtering form a data parallel single instruction multiple data (SIMD) algorithm and can be partitioned into M *CE*s. Thus, if there are N particles, ($1 < M < N$) each *CE* performs the same operations in time on $N_n = N/M$ different particles where both M and N_n are integers. An overview of this distributed particle filter architecture is given in Figure 2. The central server performs resampling — partial or full depending on the resampling scheme — and particle routing as well as overall control.

Resampling is a critical step in particle filter implementations. Without it the variance of the particle weights quickly increases, i.e., very few normalized weights remain substantial. This causes degradation in inference because the effective number of particles used for the state representation decreases. Resampling removes particle trajectories with small weights and replicates trajectories with large weights. Unfortunately, most resampling algorithms are essentially sequential. Since resampling involves updating of weights, most resampling algorithms require the completion of sampling and weight updating prohibiting concurrency between steps. Though, various efforts to derive distributed versions of resampling algorithms have been made, it has not been possible till now

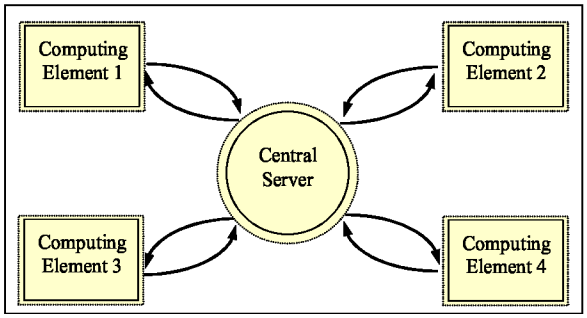


Figure 2 – System architecture for particle filter based prognostics system.

to formulate a fully distributed version. Besides, the following problems diminish the advantages of existing algorithms for partial distribution of the resampling task: (a) communication of new particles and weights amongst CEs after resampling is extensive, and (b) connections among the CEs are not known before the run-time and may change after each sampling period. Furthermore, the resampling technique used is often dictated by the application and system requirements and hence existing distributed versions cannot be used in all designs.

In the following subsections, a discussion of existing resampling techniques suitable for our domain is discussed along with the performance trade-offs involved in their use. Finally, a discussion our proposed new resampling technique is presented and discussed.

5. DISTRIBUTED RESAMPLING

In order to prevent degradation of inference in the particle filter, resampling ensures that the effective number of particles does not decrease over time. Here, effective number of particles represents the total number of statistically significant particles. It is not desirable to waste computing resources on propagating and updating particles with negligible weights. The effective number of particles in any given population is calculated as:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w_k^{(i)}\right)^2}. \quad (3)$$

Conventional Resampling Techniques

A communication-efficient conventional resampling scheme is *threshold-based resampling*. In such a scheme, when the effective number of particles is less than some threshold (i.e. $N_{eff} < N_{thr}$), resampling is performed. The new population of particles $\{\mathbf{x}_k^{(i)*}: i=1,\ldots,N\}$ is generated by sampling with replacement N times from the approximate discrete representation of the posterior state distribution given by:

$$p(\mathbf{x}_k \mid \mathbf{y}_k) \approx \sum_{i=1}^{N} w_k^{(i)} \delta\left(\mathbf{x}_k - \mathbf{x}_k^{(i)}\right), \tag{4}$$

such that $p(\mathbf{x}_k^{(i)*} = \mathbf{x}_k^{(i)}) = w_k^{(i)}$. The resampled population is independently and identically distributed with uniform weight of $1/N$.

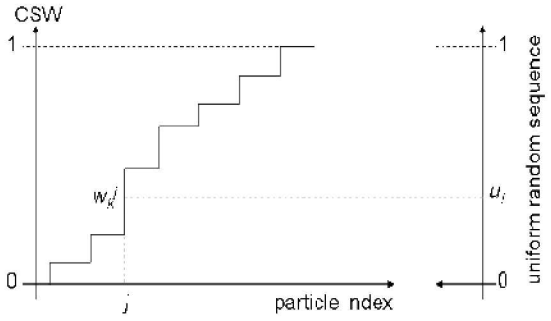


Figure 3 – Systematic resampling.

Figure 3 shows a schematic of the resampling process (CSW stands for the cumulative sum of the weights). The random variable u_j, $j=1,\ldots,N$ is uniformly distributed on [0, 1].

In terms of computing steps, first the cumulative sum of weights (CSW) of sampled particles (w_k^j for jth particle in the kth filter iteration) is computed. Then, as shown on the right of the figure, u_j (a uniform random number) is systematically updated and compared with the CSW of the particles. The particle with CSW greater than u_j is replicated. If number of particles in N, u_j is sampled N times to complete the resampling. Note that these resampling steps are carried out in the central server (Figure 2) after it has collected the updated particle values and their weights.

The value of N_{thr} chosen determines the frequency of the resampling step. This choice governs the tradeoff between prediction uncertainty (width of the state pdf) and computational and communication burden. If N_{thr} is chosen to be N, then resampling is performed every filter iteration. This resampling scheme is also known as systematic resampling. For the purposes of this paper we call this the *baseline resampling* scheme. We compare the results obtained by this method with those derived from taking $N_{thr} = 0.4 \times N$.

Parametric Resampling

In this section, we present a new resampling scheme that aims to reduce both the communication message length and the number of messages. As, shown in the results, this leads to a significant decrease in communication load. In the *threshold-based resampling* scheme, minimizing communication requirements corresponds to lowering the value of N_{thr} would achieve that. But this can correspondingly increase the degeneracy of the particle weights between resampling steps. This might lead to unacceptable uncertainty bounds for the state estimates.

Since the communication overhead of resampling stems from the need to aggregate all particle values and weights, the load may be somewhat reduced by performing resampling locally at each *CE* for most iterations and resampling globally (across all *CE*s) every few iterations. The main issue for this scheme is to maintain the statistical invariance property of resampling while doing so locally (i.e. ensuring that the statistical properties of the particle population after local resampling is unchanged). This is achieved in two ways:

1. Each *CE* operates on a statistically significant number of particles, i.e. $N_n >> 1$, where N_n denotes the number of particles for CE_n, ($\Sigma N_n = N$). Without loss of generality we assume that for all *CE*s $N_n = N/M$, where M is the number of *CE*s.
2. Any given CE_n has a particle population $\{(w_k^{(i_n)}, \mathbf{x}_k^{(i_n)}): i_n=1,\ldots,N_n\}$ representing the full state pdf. To ensure this, we perform a parametric approximation of the state pdf at the global resampling step. A mixture of Gaussians is fitted to the particle population of each CE using a least squares method. Thus for CE_n we would obtain the following vectors,
   - $\boldsymbol{\mu}_k^n$ containing the means $\mu_{j,k}^n$ of the j Gaussian kernels fitted to the population $(w_k^{(i_n)}, \mathbf{x}_k^{(i_n)})$,
   - $\boldsymbol{\sigma}_k^n$ containing the standard deviations σj,kn of the kernels, and
   - $\boldsymbol{\alpha}_k^n$ containing the relative weights $\alpha_{j,k}^n$ of the kernels such that $\Sigma_j(\alpha_{j,k}^n) = 1$.
3. The parametric estimates of all the *CE*s are communicated globally, following which the weighted sum of all the Gaussian kernels is sampled N_n times to generate the resampled population.

We call this technique *parametric resampling*. Thus, this method attempts to decrease the communication message length by representing the state pdf by 3 parameters as compared to N particles. However, this new scheme also entails additional computations for parameterizing the pdf as well as reconstruction of pdf from the paremeters. In the results for this method presented in the next section, we perform *threshold-based resampling* locally and every 3 iterations we resample globally according to the steps described above. With reference to the distributed the particle filter architecture of Figure 2, the global resampling step is performed in the central server after the individual *CE*s communicate the parametric estimates to the server.

6. EXPERIMENTS AND RESULTS

Application: Battery Health Monitoring

The application domain towards which this work is geared is battery health monitoring. Batteries form a core component of the power supply system for many machines, and their degradation often leads to reduced performance, operational impairment and even catastrophic failure. Thus, robust RUL estimation algorithms for batteries are an important research domain in prognostics. The battery aging data used in the experiments were collected from second generation 18650-size lithium-ion cells (i.e., Gen 2 cells) that were cycle-life tested at the Idaho National Laboratory under the Advanced Technology Development (ATD) Program. The battery model used in the particle filter based prognostic algorithm is shown in Figure 4.

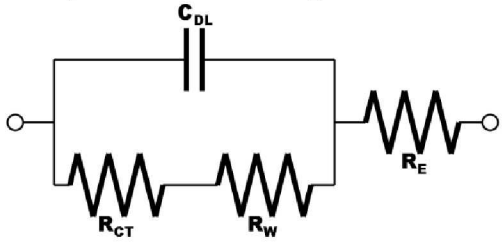


Figure 4 – Lumped Parameter Model of a Battery (revised from [17] Fig. 1).

The parameters of interest are the double layer capacitance C_{DL}, the charge transfer resistance R_{CT}, the Warburg impedance R_W and the electrolyte resistance R_E, whose values change with various aging and fault processes. From the aging data collected, R_E and R_{CT} are found to change significantly in value, and hence, are considered to be the state variables of interest. Exponential growth models, as shown in equation 5, are fitted onto their aging curves to identify the relevant decay parameters C and λ:

$$\tilde{\theta} = C \cdot \exp(\lambda t) \qquad (5)$$

where, $\tilde{\theta}$ is the model predicted value of R_E or R_{CT}. The state and measurement equations that describe the battery model are given below:

$$\begin{aligned} \mathbf{z}_0 &= \mathbf{C};\ \mathbf{\Lambda}_0 = \mathbf{\Lambda} \\ \mathbf{z}_k &= \mathbf{z}_{k-1}.\exp\mathbf{\Lambda}_k + \boldsymbol{\omega}_k \\ \mathbf{\Lambda}_k &= \mathbf{\Lambda}_{k-1} + \boldsymbol{\nu}_k \\ \mathbf{x}_k &= [\mathbf{z}_k\ ;\ \mathbf{\Lambda}_k] \\ \mathbf{y}_k &= \mathbf{z}_k + \boldsymbol{\upsilon}_k \end{aligned} \qquad (6)$$

where, the vector $\mathbf{z}$ comprises R_E and R_{CT}, and $\mathbf{C}$ and $\mathbf{\Lambda}$ contain their C and λ values, respectively. The $\mathbf{z}$ and $\mathbf{\Lambda}$ vectors are combined to form the state vector $\mathbf{x}$. The measurement vector $\mathbf{y}$ comprises the battery parameters inferred from the test data. The noise samples $\boldsymbol{\omega}$, $\boldsymbol{\nu}$ and $\boldsymbol{\upsilon}$ are picked from zero mean Gaussian distributions whose standard deviations are derived from the training data. The particle filter uses the parameterized model described in equation (6) for the propagation of the particles (samples from the pdf of $\mathbf{x}_k$). For further details, the reader is referred to [17].

Experiment Details and Results

The system architecture used is the same as outlined in Figure 2. The *CE* in our experiments is the Sun SPOT device. The resampling schemes along with their corresponding communication models were designed and simulated in MATLAB (version 8a). The times spent in message sending and receiving were measured using the SPOT device and the corresponding values were integrated into the simulation model. The software development for the Sun SPOTs was done using Netbeans IDE version 5.0. Besides the new scheme proposed in this paper, simulations for the *baseline* and *threshold-based* resampling schemes were also performed for comparison. Simulation and measurement of communication time were carried out for 2, 3 and 4 SPOT devices. For the case of 2 SPOT devices, one of them acts as the central server while the other is a CE. In all the 3 cases, the central server also performs sampling and weight updating.

Each state was a 2-dimensional vector. Also, parameter identification was done along with state estimation, hence 2-dimensional parameter values were also part of the particle along with the state values. The number of particles used was 100. Prognostics is performed by first carrying out state tracking for a few iterations followed by computation of the RUL or time-to-failure (TTF).

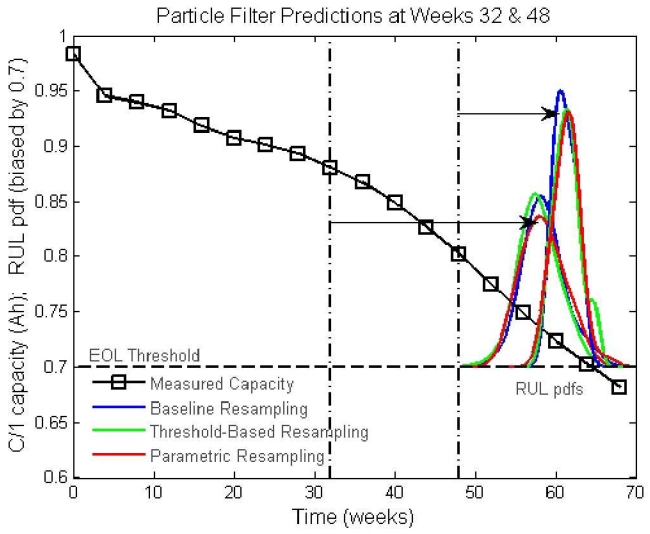


Figure 5 – Particle filter prediction pdfs for 3 resampling schemes.

Figure 5 shows the prediction pdfs for the 3 different resampling schemes for the case of distribution over 2 CEs. Two test cases for prediction were used; for the first case tracking was performed till week 32 while for the second case tracking was done till week 48 after which tracking and RUL estimation were carried out. The original dataset

contained aging data till week 72. From Figure 5 it may be observed that the RUL pdf improves in both accuracy (centering of the pdf over the actual failure point) and precision (spread of the pdf over time) for prediction at 48 weeks compared to prediction at 32 week as more measurements are included before prediction. The actual RUL for the given data set was at 64.4563 weeks.

The comparison of communication performance for the 3 resampling schemed for prediction at 32 weeks is shown in Figure 6 while the same for prediction at 48 weeks is shown in Figure 7. The communication time increases as the number of CEs is increases, since this causes an increase in the number of messages being sent. With more CEs, the number of particles being handled by a single CE decreases, however the total number of particles that needs to be distributed by the central server increases. Thus, for 2CEs, the central server needs to send only 50 particles to the other CE (each of them operate on 50 particles), but for 4CEs 75 particle values need to be sent out to the remaining 3 CEs as each of them now handle 25 particles each.

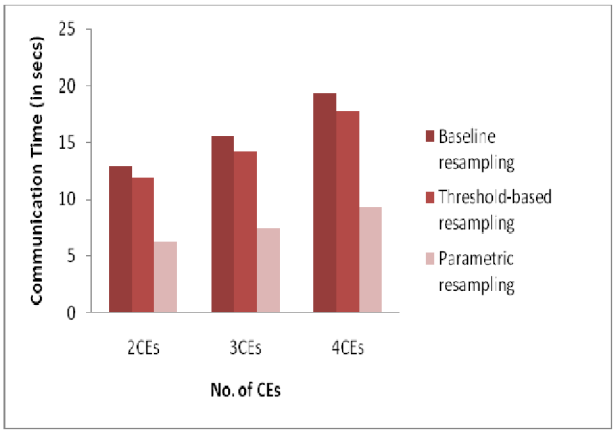


Figure 6 – Comparison of communication times for prediction at 32 weeks.

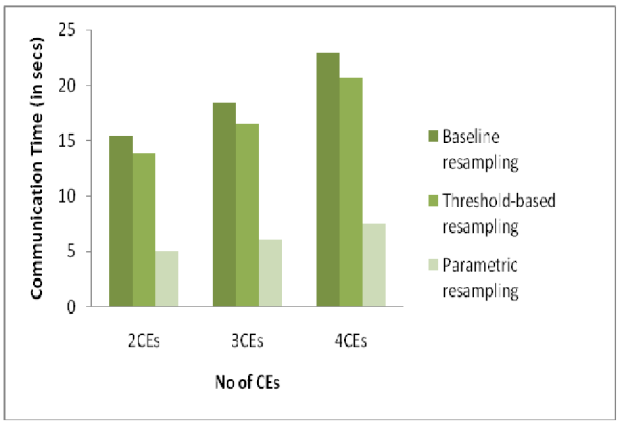


Figure 7 – Comparison of communication times for prediction at 48 weeks.

The results clearly show that parametric resampling out performs the remaining two resampling schemes by a wide margin. However, this improvement in communication also results in increase in computation as now additional computations parameterizing the pdf at each *CE* and recreation of pdf from parameters at the *central server* will have to be performed. The effects of this increase in computation as well as efficient algorithms to perform the computations are an important direction for future studies. Note that the communication times for prediction at 32 weeks are slightly less compared to the corresponding results for prediction at 48 weeks. This is due to the fact that the prediction part of the algorithm does not involve resampling and hence does not encounter performance degradation due to serialization. Thus more tracking results in more communication and hence higher execution time. The choice of how long tracking should be done is clearly and important design trade-off issue as it significantly impacts the accuracy of the prediction.

7. CONCLUSIONS

A new resampling scheme for distributed implementation of particle filters has been discussed in this paper. Analysis and comparison of this new scheme with existing resampling schemes in the context for minimizing communication overhead have also been discussed. Our proposed new resampling scheme performs significantly better compared to other schemes by attempting to reduce both the communication message length as well as number total communication messages exchanged while not compromising prediction accuracy and precision.

Future work will explore the effects of the new resampling scheme in the overall computational performance of the whole system as well as full implementation of the new schemes on the Sun SPOT devices. Exploring different network architectures for efficient communication is an importance future research direction as well.

REFERENCES

[1] M. Roemer, C. Byington, G. Kacprzynski and G. Vachtsevanos, "An Overview of Selected Prognostic Technologies with Reference to an Integrated PHM Architecture", In Proc. of the First Intl. Forum on Integrated System Health Engineering and Management in Aerospace, 2005.

[2] J. Schmalzel, F. Figueroa, J. Morris, S. Mandayam and R. Polikar, "An Architecture for Intelligent Systems Based on Smart Sensors", IEEE Transactions on Instrumentation and Measurement, Vol. 54, No. 4, Aug. 2005, pp. 1612-1616.

[3] M. Prokopenko, P. Wang, D. C. Price, P. Valencia, M. Foreman, A. J. Farmer, "Self-organizing Hierarchies in Sensor and Communication Networks". Artificial Life, Special Issue on Dynamic Hierarchies, Vol. 11(4), 407-426, 2005.

[4] A. S. Bashi, V. P. Jilkov, X. R. Li and H. Chen, "Distributed Implementations of Particle Filters", in Proc. Of Sixth International Conference of Information Fusion, 2003. Volume: 2, pp: 1164- 1171.

[5] S. Saha, C. Shen, C. Hsu, A. Veeraraghavan, G. Aggarwal, A. Sussman and S. S. Bhattacharyya, "Model-based OpenMP Implementation of a 3D Facial Pose Tracking System", in Proc. of the Wkshp. on Parallel and Distributed Multimedia, Columbus, Ohio, Aug. 2006, pp. 66-73.

[6] M. Bolic, P. M. Djuric and S. Hong, "Resampling Algorithms and Architectures for Distributed Particle Filters", IEEE Transactions on Signal Processing, Vol. 53, Issue: 7 pp. 2442- 2450 July 2005.

[7] M. Coates, "Distributed Particle Filters for Sensor Networks", in Third Intl. Symp. on Information Processing in Sensor Networks, 2004, pp. 99- 107.

[8] G. Ing and M. J. Coates, "Parallel Particle Filters for Tracking in Wireless Sensor Networks", In IEEE 6th Workshop on Signal Processing Advances in Wireless Communications, 5-8 Jun. 2005, pp. 935- 939.

[9] X. Sheng, Y.-H. Hu and P. Ramanathan, "Distributed Particle Filter with GMM Approximation for Multiple Targets Localization and Tracking in Wireless Sensor Network", in Fourth Intl. Symp. on Information Processing in Sensor Networks, 2005, pp. 181- 188.

[10] M. Rosencrantz, G. Gordon and S. Thrun, "Decentralized Sensor Fusion with Distributed Particle Filters", in Proc. Conf. Uncertainty in Artificial Intelligence Acapulco, Mexico, Aug.2003.

[11] J. L. Williams, J. W. Fisher and A. S. Willsky, "Approximate Dynamic Programming for Communication-Constrained Sensor Network Management", IEEE Transactions on Signal Processing, Vol. 55, Issue 8, Aug. 2007, pp. 4300-4311.

[12] S. S Iyengar, Q. Wu, N. S. V. Rao, "Networking paradigm for distributed sensor networks", In Proc. of the Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Sept. 8-10, 2003, pp. 284-290.

[13] W. Wang, S. Lafortune and F. Lin, "A Polynomial Algorithm for Minimizing Communication in a Distributed Discrete Event System with a Central Station", In Proc. Of 45th IEEE Conference on Decision & Control, San Diego, CA, USA, December 13-15, 2006.

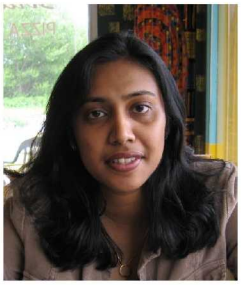
[14] S. Saha, B. Saha and K. Goebel, "Distributed Prognostics Using Wireless Embedded Devices". In IEEE International Conference on Prognostics and Health Management, Denver, CO, USA, October 2008.

[15] S. Arulampalam, S Maskell, N. J Gordon and T. Clapp, "A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking", IEEE Trans. on Signal Processing, vol. 50, no. 2, pp. 174-188, 2002.

[16] N. J. Gordon, D. J. Salmond and A. F. M. Smith, "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation", Radar and Signal Processing, IEE Proceedings F, vol. 140, no. 2, pp. 107-113, April 1993.

[17] B. Saha and K. Goebel, "Uncertainty Management for Diagnostics and Prognostics of Batteries using Bayesian Techniques", in Proc. 2008 IEEE Aerospace Conference, March 2008.

BIOGRAPHIES



Sankalita Saha received her B. Tech (Bachelor of Technology) degree in Electronics and Electrical Communication Engineering from Indian Institute of Technology, Kharagpur, India in 2002 and Ph.D. degree in Electrical and Computer Engineering from University of Maryland, College Park in 2007. She is currently a post-doctoral scientist working at RIACS/NASA Ames Research Center, Moffett Field, CA. Her research interests are in prognostics algorithms and architecture, distributed systems and system synthesis.

***Bhaskar Saha** is a Research Programmer with Mission Critical Technologies at the Prognostics Center of Excellence, NASA Ames Research Center. His research is focused on applying various classification, regression and state estimation techniques for predicting remaining useful life of systems and their components. He has also published a fair number of papers on these topics. Bhaskar completed his PhD from the School of Electrical and Computer Engineering at Georgia Institute of Technology in 2008. He received his MS from the same school and his B. Tech. (Bachelor of Technology) degree from the Department of Electrical Engineering, Indian Institute of Technology, Kharagpur.*

Kai Goebel *is a senior scientist at NASA Ames Research Center where he leads the Prognostics Center of Excellence (prognostics.arc.nasa.gov). Prior to that, he worked at General Electric's Global Research Center in Niskayuna, NY from 1997 to 2006 as a senior research scientist. He has carried out applied research in the areas of artificial intelligence, soft computing, and information fusion. His research interest lies in advancing these techniques for real time monitoring, diagnostics, and prognostics. He has fielded numerous applications for aircraft engines, transportation systems, medical systems, and manufacturing systems. He holds half a dozen patents and has published more than 75 papers. He received the degree of Diplom-Ingenieur from the Technische Universität München, Germany in 1990. He received the M.S. and Ph.D. from the University of California at Berkeley in 1993 and 1996, respectively.*